CHAPTER 4
CONCEPT LATTICE MODEL

4.1 INTRODUCTION

Concept lattices were introduced more than 50 years ago in Birkhoff’s first book on lattice theory. More than 20 years ago, Ganter and Wille started to expand the theory considerably and investigated serious applications of concept analysis e.g. in the social sciences. But only ten years ago did a few researchers start to explore the possibilities of concept lattices for computer science, in particular software technology.

In mathematical terms, a concept lattice is a partially ordered set in which any two elements have a unique greatest lower bound and a least upper bound. The concept lattices can be used in web services for clustering and discovery. The lattices are represented using a graph in the computer program and they can be used for search optimization problems. If a problem is represented in the form of lattices, then the search can be performed with ease. The concept lattices can also figure out the inherent relation between the items in a set. The concept lattices are built using the web service operations and they can be pivotal in discovering the necessary backup services for the composite web applications.
4.2 SYSTEM MODEL

The system is designed in such a way that it addresses the goals and objectives of the research work described in the earlier sections. Figure 4.1 depicts the detailed design of the proposed system and the components used in the architecture are described in the following discussion.

![Generalized System Architecture](image)

**Figure 4.1 Generalized System Architecture**

The *service provider* is the entity that has the web service and the service repository is maintained by the service provider. The service provider is termed as the actual owner of the web service and he registers the service to the UDDI which can be used by the service requestors. The *business interface* is used by the service providers for registering them as a service provider and to register the available
service to the service registry. The business interface receives the service
descriptions in the form of a WSDL document and the functionalities are obtained
from the service descriptions.

The *WSDL parser* is used to parse the WSDL file for finding the service
name, descriptions, operations and the end point of the web service. The data
obtained from the WSDL parser are tabulated in the UDDI Business Registry (UBR),
which is then looked up during the discovery process. The WSDL parser implements
the Algorithm 1 presented below that can be viewed as XML parsing.

**Algorithm 1** WSDL Parser

<table>
<thead>
<tr>
<th>In:</th>
<th>Service Descriptions $W_{ws}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out:</td>
<td>Service Name $N_{ws}$, Operations $O_{ws}$ and Location $L_{ws}$</td>
</tr>
</tbody>
</table>

$O_{ws} = \text{null}$

*for each service* $ws$ *in* $W_{ws}$

*get Service Name* $N_{ws}$

*for each port* $p$ *in* $ws$

*get Service Location* $L_{ws}$

*get PortName, PortType*

*for each Operation* $op$ *in* $PortType$

$O_{ws} = O_{ws} \cup \text{OperationName}$

*end*

*end*

*end*
The UDDI Business Registry (UBR) is responsible for creating the registry of web services. The service provider publishes the web service using the WSDL file along with the functional descriptions which are parsed by the WSDL Parser component and the service is registered with the UBR with its own description. The web service interface is used by the service requestor or the client to search the services available in the service registry. The web service interface is the front end and the client communicates to the service registry using this component.

The above Clustering Module clearly explains the process of the formation of the lattice structure. The registered service by the service provider in the business registry is parsed by using the parsing algorithm and it finds the operations of the services based on their descriptions and then a matrix form is generated. It is called as SimMat and based on this SimCtx is generated and thus the lattice is formed.

4.3 STRUCTURE OF THE LATTICE NODE

The following diagram gives a sketch of the structure of the Lattice Mode. When read together with the adjoining exemplification, it makes the structure very clear.
The terms in the Figure 4.2 are exemplified below,

- **ws_id** is the unique *web service identifier* for easy identification and indexing of web service in the UDDI Business Registry. The search can be easily performed using the numbers as searching and matching a string is more problematic than searching numbers.

- **ws_na** is the *web service name* provided by the service provider and it offers the option to map web service requests based on the qualified name. In terms of the operation, the name attributes of the input and output elements provide a unique name among all input and output.

- **ws_op** is the *web service operation* which is provided by the web service, and mainly concludes the publication of service descriptions, lookup or finding of
service descriptions, and binding or invoking of service based on the service description.

- **wsdl** is the *web services description language* and it provides a machine readable description of how the service can be called, what parameter it expects and what data structure it returns.

- **Resp** is referred to as the *Response Time*, it is mainly based on the time duration between a service user sending a request and receiving the corresponding response. It is measured in millisecond.

- **Avty** is referred to as the *Availability*, it is the ratio of the number of successful invocations to total invocations. It is measured in percentage.

- **Thpt** is referred to as the *Throughput*, which is the total number of invocations for a given period of time. It is measured in invocations per second.

- **Rety** is referred to as the *Reliability*, which is the ratio of the number of error messages to total messages. It is measured in percentage.

- **LatT** is referred to as the *Latency*, which is the time taken for the server to process a given request. It is measured in milliseconds.
- *Relv* is referred to as the *Relevance*, which is the rank of web service quality and is measured in percentage.

- *Rate* is referred to as the *Rating*, it is based on the user feedback and is updated every time the user gives rating to the service.

A node in the lattice structure can be defined as a set of fields as,

$$L_n = \begin{cases} 
(SN, SM) \\
SN = (SN_1, SN_2, \ldots, SN_n), \\
SM = (SMQ, SMU), \\
SMQ = (SMQ_1, SMQ_2, \ldots, SMQ_k), \\
SMU = (SMU_1, SMU_2, \ldots, SMU_k),
\end{cases}$$ where, $0 < i \leq n$, $0 < j \leq k$, $0 < k \leq u$.

where,

- $L_n$ is the Lattice node structure,

- SN refers to the Syntactic Information and ‘$SN_i$’ is a particular attribute ‘i’ and ‘n’ is the number of fields to accommodate the syntactic information where $0 < i \leq n$. Since this node structure offers a *scalar type data structure*, i.e. one
field can accommodate only one type of information, ‘n’ is the number of fields and types of information stored under Syntactic Head.

- SM refers to the Semantic Information and SMQ refers to the QoS information under Semantic Head and SMU refers to the user’s feedback information under Semantic Head.

- ‘SMQ_j’ is a particular attribute ‘j’ and ‘q’ is the number of fields to accommodate the semantic information where $0 < j \leq q$. Since this node structure offers a scalar type data structure, i.e. one field can accommodate only one type of information, ‘q’ is the number of fields and types of information stored under Semantic QoS Head.

- ‘SMU_k’ is a particular attribute ‘k’ and ‘u’ is the number of fields to accommodate the semantic information where, $0 < k \leq u$. Since this node structure offers a scalar type data structure, i.e. one field can accommodate only one type of information, ‘u’ is the number of fields and types of information stored under Semantic User’s Feedback Head.
4.4 CONCEPT LATTICE

The purpose of Formal Concept Analysis or FCA is to automatically find groups of objects (or entities) that share in common a group of attributes. It is very natural to apply concept lattices for software analysis, as every software artefact contains an abundance of relations between "objects" and "attributes". To explore hidden structure in such relations is a natural task whenever one wants to understand old software artefacts, or reengineer legacy systems. As a result, a wave of concept lattice applications in software technology was proposed. Some of the applications were well-motivated and based in a thorough understanding of the underlying theory, while others just generated lattices from "yet another relations", without validating the resulting structures.

Formal Concept Analysis is the principled way of deriving a concept hierarchy or formal ontology from a collection of objects and their properties. Each concept in the hierarchy represents the set of objects sharing the same values for a certain set of properties and each sub-concept in the hierarchy contains a subset of the objects in the concepts.

In this section, an approach using some basic formal definitions, along with an illustrative example is used. For clarity's sake, the approach using an imaginary set of web services for performing calculations is illustrated. Each service from this set is parsed by a WSDL parser to extract its signatures. The set of services with their signatures are given unique identifiers, as listed in table 4.2 [10]
A similarity measure is chosen, and the operations signatures extracted from the WSDL files are used by this similarity measure according to its input format. Several similarity measures for web services exist in the literature.

<table>
<thead>
<tr>
<th>Services</th>
<th>Id</th>
<th>Operations</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile</td>
<td>ws₁</td>
<td>GetCost, GetModel</td>
<td>op₁₁, op₁₂</td>
</tr>
<tr>
<td>Sports</td>
<td>ws₂</td>
<td>GetBall</td>
<td>op₂₁</td>
</tr>
<tr>
<td>Shipping</td>
<td>ws₃</td>
<td>Cargo, GetTicket, GetLocation, GetPassenger</td>
<td>op₃₁, op₃₂, op₃₃, op₃₄</td>
</tr>
</tbody>
</table>

The similarity according to the semantics is evaluated and the similarity measure is applied on various pairs of operations provided by different services. The similarity between operations provided by the same service is not considered because when a service becomes dysfunctional, all of its operations become dysfunctional too. The similarity is assessed in the form of values in the range [0, 1]. If two operations are sufficiently similar, the similarity value will approach 1, or else it will approach 0.

A Similarity measure \((Sim)\) can be defined as follows: [12]

\[
Sim : O \times O \rightarrow [0, 1]
\]
\[ \forall op_{ij} \in O \rightarrow Sim(op_{ij}, op_{ij}) = 1 \]

(an operation with itself)

\[ \forall op_{ij} \neq op_{ik} \in O \rightarrow Sim(op_{ij}, op_{ik}) = 0 \]

(operations in the same service)

\[ \forall op_{ij}, op_{nm} \in O \rightarrow Sim(op_{ij}, op_{nm}) [0,1] \]

(operations in different services)

The similarity values that are calculated can be presented with a symmetric square matrix (SimMat), as shown in table 4.2. This matrix is of size \( n=|O| \) and its diagonal elements are mostly equal to 1, since the similarity of an operation with itself is equal to, 1 as mentioned above. From the similarity matrix SimMat, we can extract several binary contexts by specifying threshold values \( \Theta \in [0,1] \). Thus, the values of SimMat that are greater or equal to the chosen threshold \( \Theta \) are scaled to 1, while other values are scaled to 0. The binary context that corresponds to \( \Theta = 0.75 \) is shown in table 4.3, It is called SimCxt. [12]

The SimCxt is a triple \((O, O, RSim_{\Theta})\), where \( RSim_{\Theta} \) is a binary relation indicating whether an operation is similar to another operation or not.

\[(op_{ij}, op_{nm}) \in RSim_{\Theta} \iff Sim(op_{ij}, op_{nm}) \geq \Theta\]

The SimCxt context is used to generate a lattice of operations, \( \beta(O, O, RSim_{\Theta}) \). This lattice helps in discovering groups of similar operations, which are used later on to construct the services lattice.
Table 4.2 The similarity matrix (\(SimMat\)) [12]

<table>
<thead>
<tr>
<th></th>
<th>(op_{11})</th>
<th>(op_{12})</th>
<th>(op_{21})</th>
<th>(op_{31})</th>
<th>(op_{32})</th>
<th>(op_{33})</th>
<th>(op_{34})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(op_{11})</td>
<td>1</td>
<td>0</td>
<td>0.75</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(op_{12})</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(op_{21})</td>
<td>0.75</td>
<td>0</td>
<td>1</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
<td>0.75</td>
</tr>
<tr>
<td>(op_{31})</td>
<td>0.5</td>
<td>0</td>
<td>0.75</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(op_{32})</td>
<td>0</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(op_{33})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(op_{34})</td>
<td>1</td>
<td>0</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3 The Binary context (\(SimCtx\)) for \(\Theta = 0.75\)

<table>
<thead>
<tr>
<th></th>
<th>(op_{11})</th>
<th>(op_{12})</th>
<th>(op_{21})</th>
<th>(op_{31})</th>
<th>(op_{32})</th>
<th>(op_{33})</th>
<th>(op_{34})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(op_{11})</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(op_{12})</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(op_{21})</td>
<td>X</td>
<td>X</td>
<td>(x)</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(op_{31})</td>
<td></td>
<td>X</td>
<td>(x)</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(op_{32})</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(op_{33})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(op_{34})</td>
<td>X</td>
<td>(x)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

In the resulting operation lattice, groups of mutually similar operations can be identified by the concepts having equal extent and intent sets. Such concepts are termed as square concepts because they form square gatherings on the binary context matrix.
In short, concept lattices can be used for clustering the web services semantically based on the operations provided by the web services. The significance of the concept lattices are mentioned above and the attributes of significance can be seen as:

- Easy to represent
- Linked using a hierarchical relation
- Takes less time to search
- Can incorporate background knowledge
The mapping model serves as a tool for depicting the importance of concept lattices for clustering of web services semantically with quality factors considered. The figure (Figure 4.4) below explains the mapping from the attributes of challenges to the attributes of concept lattices.

4.5 MAPPING MODEL

Mapping Model describes the attributes of the challenges and the basic functionalities that help to solve them have been read out from the existing researchers. They have been helpful in identifying the most appropriate approach to solve the scenario.

Figure 4.4 Attribute Mapping Model
The primary attribute of the challenge is to represent the functionality of the web services and it can be easily solved by lattices as they are easy to represent. The concept lattices are linked using a hierarchical relationship, which enables to identify the similar services in less time. Though concept lattices fail to address QoS constraints, this research makes use of them by containing the parameters in the node structure of the concept lattices.

4.6 SUMMARY

This describes the concept lattices and their significance, and briefly outlines their adoption and the overall system design. In mathematical terms, a concept lattice is a partially ordered set in which any two elements have a unique greatest lower bound and a least upper bound. The concept lattices can be used in web services for clustering and discovery. The lattices are represented using a graph in the computer program and they can be used for search optimization problems. If a problem is represented in the form of lattices, then the search can be performed with ease. The concept lattices can also figure out the inherent relation between the items in a set.

The registered service by the service provider in the business registry is parsed by using the parsing algorithm and it finds the operations of the services based on their descriptions and then a matrix form is generated. It is called as SimMat and based on this SimCtx is generated and thus the lattice is formed.
Formal Concept Analysis is the principled way of deriving a concept hierarchy or formal ontology from a collection of objects and their properties. Each concept in the hierarchy represents the set of objects sharing the same values for a certain set of properties and each sub-concept in the hierarchy contains a subset of the objects in the concepts.

Concept lattices can be used for clustering the web services semantically based on the operations provided by the web services. The significance of concept lattices is that they are linked using a hierarchical relationship, which enables to identify the similar services in less time. Though concept lattices fail to address QoS constraints, this research makes use of them by containing the parameters in the node structure of the concept lattices.